REMARKS

I. Overview

These remarks are set forth in response to the Last Non-Final Office Action. Presently, claims 1 through 14 are pending in the Patent Application. Claims 1 and 9 are independent in nature. In the Last Non-Final Office Action, claims 9 through 14 have been rejected under 35 U.S.C. § 101. Further, claims 1 through 14 have been rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,947,943 to DeAnna et al. ("DeAnna"). In response, *Applicants have amended claim 9 to address the rejections under 35 U.S.C. § 101, but otherwise Applicants respectfully traverse the rejections on the art.*

II. The Applicants' Invention

As set forth in the first paragraph of page 9 of Applicants' originally filed specification, the Applicants have invented an IMAP server which has been programmed for integration with a collaborative messaging application within an application server. The IMAP server can include a platform independent collection of classes and can be configured to operate within a virtual machine. To that end, the IMAP server can be a Java application

which comports with the J2EE specification. The IMAP server can be communicatively coupled to a data store of messages and can respond to requests to manage the messages through the collaborative messaging application. Additionally, access to the IMAP server can be regulated by authentication logic disposed within the application server.

III. Rejections Under 35 U.S.C. § 101

At page 2 of the Last Non-Final Office Action, Examiner rejects system claims 9 through 14 as being directed to non-statutory subject matter. With respect to Applicants' system claim 9, as set forth in M.P.E.P. 2106.01, subpart I, as part of an analysis of a system claim, an Examiner should determine whether a claim incorporating a computer program has been claimed as part of an otherwise statutory manufacture or machine. In such a case, the claim remains statutory irrespective of the fact that a computer program is included in the claim. The same result occurs when a computer program is used in a computerized process where the computer executes the instructions set forth in the computer program. Only when the claimed invention taken as a whole is directed to a mere program listing, i.e., to only its description or expression, is it descriptive material *per se* and hence non-statutory.

To that end, as expressed in M.P.E.P. 2106.01, since a computer program is merely a set of instructions capable of being executed by a computer, the computer program itself is not a process but when a computer program is claimed in a process where the computer is executing the computer program's instructions, the Examiner should treat the claim as a process claim. Conversely, when a computer program is recited in conjunction with a physical structure, such as a computer memory, the Examiner should treat the claim as a product claim. In the instant case, Applicants' system claims indicate that the logic of a computer program is operable and performs functions, necessarily the logic executes by a processor in the memory of a computer. Accordingly, Examiner should treat Applicants' system claims as a product under M.P.E.P. 2106.01.

IV. Rejections Under 35 U.S.C. § 102(e)

At pages 3 and 4 of the Last Non-Final Office Action, Examiner generally rejects claims 1 through 14 and specifically claim 1 as being anticipated by DeAnna. Under the law, the factual determination of anticipation under 35 U.S.C. § 102 requires the identical disclosure, either explicitly or inherently, of each element of a claimed invention in a single

reference.[1] Moreover, the anticipating prior art reference must describe the recited invention with sufficient clarity and detail to establish that the claimed limitations existed in the prior art and that such existence would be recognized by one having ordinary skill in the art.[2] Absence from an allegedly anticipating prior art reference of any claimed element negates anticipation.[3]

With the law of anticipation in mind, claims 1 and 9 relate to a mail server cell for collaborative messaging. Exemplary claim 1 recites:

> 1. In a collaborative messaging system, a mail server cell comprising:
> a logical grouping of application server nodes disposed and executing within an application server hosting an application programming interface to expose business logic and business processes for use by other applications;
> an Internet Message Access Protocol (IMAP) compliant mail server executing in a computer and coupled to said logical grouping of application server nodes; and,
> at least one data store configured for storing electronic mail messages processed in said IMAP compliant mail server.

---

[1] In re Schreiber, 128 F.3d 1473, 1477 (Fed. Cir. 1997) ("To anticipate a claim, a prior art reference must disclose every limitation of the claimed invention, either explicitly or inherently"), In re Rijckaert, 9 F.3d 1531, 28 USPQ2d 1955 (Fed. Cir. 1993); Richardson v. Suzuki Motor Co., 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989); Perkin-Elmer Corp. v. Computervision Corp., 732 F.2d 888, 894, 221 USPQ 669, 673 (Fed. Cir. 1984).

[2] See In re Spada, 911 F.2d 705, 708, 15 USPQ 1655, 1657 (Fed. Cir. 1990); Diversitech Corp. v. Century Steps Inc., 850 F.2d 675, 678, 7 USPQ2d 1315, 1317 (Fed. Cir. 1988).

[3] Kloster Speedsteel AB v. Crucible, Inc., 793 F.2d 1565, 1571 (Fed. Cir. 1986)(emphasis added).

Integral to claim 1, and also claim 9 which recites similar operable limitations, is the presence of an application server hosting an application programming interface to expose business logic and business processes for use by other applications. So much cannot be found in DeAnna.

Notwithstanding, Examiner argues to the contrary at page 3 of the Last Non-Final Office Action. Specifically, Examiner states::

> As to claim 1, Anna teaches a collaborative messaging system, a mail server cell comprising:
> a logical grouping of application server nodes disposed within an application server (see col. 5 lines 46-60, col. 7 lines 7-35 and col. 3 lines 64-col. 4 lines 7, the Zeospehere server has a plurality of applications for communicating with a plurality of servers and devices) hosting an application programming interface to expose business logic and business process for use by other applications (see col. 17 lines 33-52, col. 18 lines 57-col. 19 lines 17 and col. 19 lines 19-35 and col. 20 lines 31-52, Zeosphere has API to handle business requests such as purchase order and account information);

Thus, Examiner has equated the claimed "application server hosting an API to expose business logic and business processes for use by other applications" with an "application hosting an API to expose business logic and business process for use by other applications" as allegedly set forth in column 17, lines 33 through 52, column 19, lines 19 through 35 and column 20, lines 31 through 52 of DeAnna. With respect, Examiner's analysis is flawed on two accounts.

First, Examiner has expressly compared an "application" to an "application server" and in doing so, has omitted consideration of the word "server". As it is extraordinarily well known in the art, an "application server" is quite different than an ordinary "application". An ordinary application can include any computer program such as a word processor, spreadsheet or computer game. In the art, however, an application server is

> An **application server** is a software framework that provides an environment where applications can run, no matter what the applications are or what they do. It is dedicated to the efficient execution of procedures (programs, routines, scripts) for supporting the construction of applications.

(Wikipedia). Second, column 17, lines 33 through 52, column 19, lines 19 through 35 and column 20, lines 31 through 52 of DeAnna fail to provide a teaching of "application server". Rather, column 17, lines 33 through 52, column 19, lines 19 through 35 and column 20, lines 31 through 52 of DeAnna provide verbatim:

> In ZeoFusion, the ProcessDefinition interface provides for the execution of a workflow process definition. A workflow process definition is the collection of statements which define a workflow process. Typically this entails the definition of a number of states through which a ProcessInstance transitions through and a set of actions that should be carried out in each of those states. Each ProcessInstance has a single ProcessDefinition, as such a ProcessInstance can follow only one DecisionFlow process. The RuleProcess provides the ability to execute workflow process definitions which are defined in rules. The rules are then executed using the inference engine which has been loaded by the Rulebean. The JavaProcess is an abstract class that represents a process

> definition created as a Java class. All process definitions that are created in Java should extend this class, defining the execute method. In the execute method the ProcessDefinition should evaluate the state of the ProcessInstance and, based upon that state, create, configure and add actions to the ProcessInstance. These actions would then be executed by the WorkflowProcessor immediately after the execute method returns.
>
> ZeoFusion's Actions, and the order in which they're carried out, are what defines a workflow process. There are seven concrete workflow actions defined: Question, Notification, Pause, JavaCommand, EventBroadcast, DatabaseAction and ScriptExecution. Actions are created, configured and then added to the ProcessInstance by the ProcessDefinition. Actions are then executed by the WorkflowProcessor, sequentially, in the order in which they were added. The Action has the capability to facilitate the repeated execution of the Action in the case that some external influence has prevented its forward movement and eventual completion. An example of this would be a Question, which requires a response from the person or persons to whom it was addressed. If the response is not received after a set period of time the question may be re-sent to the same individual to further solicit a response. This process would be repeated until the maximum attempts value has been exceeded.
>
> 4. The Script action: a class designed to execute an external script in any of the languages supported by the IBM bean Scripting Framework (BSF), an architecture for incorporating scripting into Java applications and applets. Scripting languages such as Netscape Rhino (Javascript), VBScript, Perl, Tcl, Python, NetRexx and Rexx are commonly used to augment an application's function or to script together a set of application components to form an application. The necessary support files are bsf.jar and bsfengines.jar for the BSF, jpython.jar for JPython support, and jacl.jar and tcljava.jar for JACL support. Other languages can be added as needed.

Thus the cited portion of DeAnna pertains not to an application server as recognized by the Honorable Board of Patent Appeals and Interferences at page 3 of the Decision on Request for Rehearing dated October 1, 2010, but to a workflow engine. Accordingly, DeAnna lacks the claimed teaching of application server hosting an API to expose business logic and business

processes for use by other applications and cannot satisfy a prima facie case of anticipation.

V. Conclusion

Applicants respectfully requests the withdrawal of the rejections under 35 U.S.C. §§ 101 and 102(e)) owing to the amended claims and foregoing remarks. The Applicant requests that the Examiner call the undersigned if clarification is needed on any matter within this Amendment, or if the Examiner believes a telephone interview would expedite the prosecution of the subject application to completion.

Respectfully submitted,

Date: July 1, 2011

/Steven M. Greenberg/

Steven M. Greenberg
Reg. No.: 44,725
Carey, Rodriguez, Greenberg & Paul
950 Peninsula Corporate Circle
Suite 2022
Boca Raton, Florida 33487
Customer No. 46321
Tel: (561) 922-3845
Fax: (561) 244-1062